SPEEDFORCE DIGITAL

Laravel Candidate Assessment

Thank you for your interest in becoming part of our team. The purpose of this challenge is to give you a feeling for the tasks that await you at SpeedForce and to familiarize you with our current stack.

The whole process should not take you longer than a few hours.

PLEASE NOTE: For us the most important aspect we are looking for in the code challenge is that you respect the guidelines and specifications given in the description. This is the only way we can identify whether you are able to follow the structure of our project in the way that we imagine it.

## Tech Stack for this assignment

- Laravel 9, PHP 8

- Bootsrap 5

- JQuery

- MySQL

Please use the given UI componentes / blade files for the frontend, and focus on the backend only with a bit of Jquery.

The project in production is very similar: A Laravel 8 monolith + Bootsrap + Jquery. In the future, however, the whole thing will probably be rewritten to a REST API + Flutter / React Native.

## Getting Started 🏃

- Create a local copy of the repository.

- Make sure you have xxamp or something similiar.

- Create a database, name it 'SpeedForce_coding_challenge'.

- Setup .env file

- Run ```composer install``` and ``` php artisan key:generate ```.

- When you see the login page, head over the register page, create an account and log in. After that you should see some of the components in the images below and you can get started in the network_connections.blade.php file.

## Objective ✅

Demo Video:

https://user-images.githubusercontent.com/76958067/170489878-41332cb5-6264-427e-a094-eeb09b8c8af4.mp4

Develop common connection features of social media platforms. In detail:

As a user:

- when I click on "Suggestions", I want to be able to see all the other users I am not yet connected to, who I have not sent an invitation to, and who have not sent me an invitation.

- When I click on "Connect" I want to send this user a request to connect.*
![grafik](https://user-images.githubusercontent.com/76958067/170004194-4f0a4962-4759-491a-817b-8d18f8b71e12.png)

- when I click on "Sent Request", I want to see all the requests I have sent to connect.

- When I click on "Withdraw Request" I want to withdraw my request.*
![grafik](https://user-images.githubusercontent.com/76958067/170004103-5c8396a1-6897-4abd-8f41-e22ece4b4991.png)

- when I click on "Received Requests", I want to see all the requests that have been sent to me

- when I click on "Accept", I want to add this user to my network.*
![grafik](https://user-images.githubusercontent.com/76958067/170003922-9b4786f6-1da3-422a-aa15-14ff3f3f4e64.png)

- when I click on "Connections", I want to see all my connections.

- when I click on "Remove Connection", I want to remove this user from my network.*

![grafik](https://user-images.githubusercontent.com/76958067/170004446-9a2d2ba8-6d73-4cc2-9075-4795dcf94bea.png)

- when I click on "Connections in common", I want to see all connections in common with that user.

![grafik](https://user-images.githubusercontent.com/76958067/170006264-c59dae03-3164-4198-9119-700a9c76f0cd.png)

Further requirements:

- The brackets should always contain the total number of the respective categories.

![grafik](https://user-images.githubusercontent.com/76958067/170007652-8dc86360-9b06-46d0-bffd-c412e928ae88.png)

- To test all of the above please include seeders and factories, to generate suggestions, requests by other users and connections in common with the command:

    DatabaseSeeder.php

    ```php artisan migrate --seed```. Connections could then be tested by accepting a request.

    ``` <?php

    namespace Database\Seeders;

    use Illuminate\Database\Seeder;

    use Database\Seeders\UsersSeeder;

    use Database\Seeders\RequestsSeeder;
    ```

```
use Database\Seeders\ConnectionsInCommonSeeder;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;


class DatabaseSeeder extends Seeder

{

    /**

     * Seed the application's database.

     *

     * @return void

     */

    public function run()

    {

        $this->call(UsersSeeder::class);

        $this->call(RequestsSeeder::class);

        $this->call(ConnectionsInCommonSeeder::class);

    }}
```

- If there are no connections in common, the button should get the class 'disabled'.

- All new routes should be written into a new routing file (e.g. userConnection.php), that must be registered inside the RouteServiceProvider.php file.

- There are blade components such as connection, request and so on. You can use all of them or decide to only use one component with if clauses.

- Before each ajax call (only for getting suggestions, request and connections), show the loading skeletons and hide them after a successfull response from the server.

- Javascript code should be written in the main.js file

*(HTML element should disappear afterwards)

## Coding and naming conventions 🚨

- Name HTML ids in snake_case.

- Javascript & PHP variables in camelCase.

- Make sure that your code is readable for others and include comments in important places. (Guideline: Code = How?, Comment = What?)

- There are a few pre-built Javascript functions in the helper.js file to help you work with AJAX. You can use them, but you don't have to. But please stick to using Jquery.

## When you are done 🎖️

- Please check again if all requirements have been done as described. If there are things that need explanation, add them as a text message via email at hr@SpeedForce.agency

- Reply to a mail containing a link to a private repository with your code or a zip file.